词法分析评审材料

本程序实现了所有要求的功能

【操作步骤】

首先进入gpl-master目录，然后输入：

make

cd sample

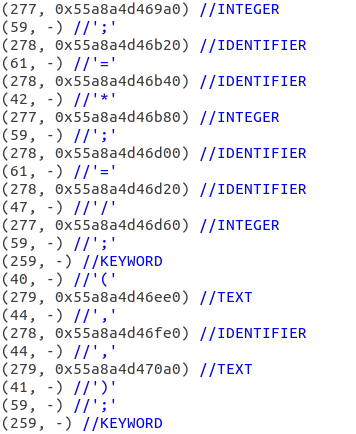
../gpl calc -lex

程序会在sample目录下输出calc.lex文件

其中就有需要输出的内容

**1.输出单词（以二元式形式）**

【运行结果】



按照老师的要求给出了输出的二元式形式，并且为了方便辨别识别到单词的具体内容，我在后面加了例如//KEYWORD //TEXT //’;’这样的注释，可以方便人阅读。

【代码简述】

主要是在yylex里面 按照老师给出的一些例子 然后用类似的方法写出要支持的功能

**2.出错处理**

【运行结果】

【用例1】非法组合出错

error1.gpl

!? //这是程序的内容 就一行!?

结果：



【说明】遇到!? !! !. !\*等非法组合时会报错

【用例2】非法字符出错

error2.gpl

我爱你 //这是程序的内容 就一行中文字符“我爱你”

结果：



【说明】遇到非法字符 如中文字符时会报错

2）部分代码与细节

【思路】在yylex识别单词的过程中

我主要在yylex中增加了以下代码处理异常：

if(character=='!')

{

concat();

getch();

if(character=='=')

{

concat();

printf("(%d, -) //NE\n", NE);

return NE;

}

else if(!digit()||!letter())

{

yyerror("invalid symbol combination");

exit(1);

}

else

{

retract();

printf("(%d, -) //'!'\n", '!');

return '!';

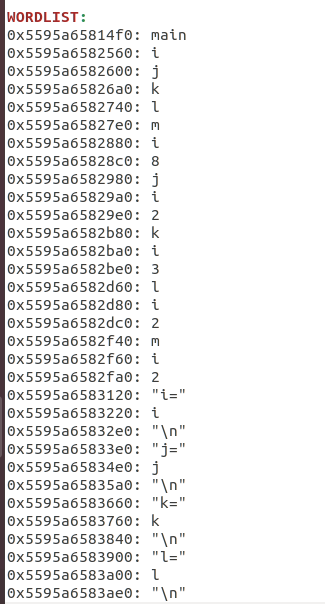
}

}

在识别’!’的同时，如果识别到!? !@ !#等非法组合，可以报错

**3.符号表管理**

【运行结果】



【思路】通过创建符号表结构体，创建全局变量，在yylex进行识别单词时，同时把相关地址和字面量存入表中，最后再输出表即可。

【详细细节与部分代码】

结构体定义如下：

typedef struct wordlist

{

int \*p;

char m[100];

}WORDLIST;

相关代码如下：

if(letter())

{

while(letter() || digit())

{

concat();

getch();

}

retract();

num=keyword();

if(num!=0)

{

printf("(%d, -) //KEYWORD\n", num);

return num; // return keyword

}

else

{

lexeme=malloc(strlen(token+1));

strcpy(lexeme,token);

yylval.string=lexeme;

wl[lenwl].p = (int\*)lexeme;

strcpy(wl[lenwl].m, token);

lenwl++;

printf("(%d, %p) //IDENTIFIER\n", IDENTIFIER, lexeme);

return IDENTIFIER;

}

}

if(digit())

{

while(digit())

{

concat();

getch();

}

retract();

lexeme=malloc(strlen(token+1));

strcpy(lexeme,token);

yylval.string=lexeme;

int \*p = malloc(4);

\*p = atoi(lexeme); // \*p is an int

wl[lenwl].p = (int\*)p;

strcpy(wl[lenwl].m, token);

lenwl++;

printf("(%d, %p) //INTEGER\n", INTEGER, p);

return INTEGER;

}

if(character=='"')

{

concat();

getch();

while(character!='"' && character!=EOF)

{

concat();

getch();

}

if(character==EOF)

{

printf("lex error: \" expected\n");

exit(1);

}

concat();

lexeme=malloc(strlen(token+1));

strcpy(lexeme,token);

yylval.string=lexeme;

wl[lenwl].p = (int\*) lexeme;

strcat(wl[lenwl].m, token);

lenwl++;

printf("(%d, %p) //TEXT\n", TEXT, lexeme);

return TEXT;

}

最后再main函数中输出词汇表即可：

if(mode==1)

{

yylex();

printf("\n\n\nWORDLIST:\n");

//printf("%p, %s", wl[0].p, wl[0].m);

int i;

for(i=0;i<lenwl;i++)

{

printf("%p: %s\n", wl[i].p, wl[i].m);

}

exit(0);

}

4.问题一：你的代码用于存放3种符号的值的代码位置在哪里

主要是通过（3）中提到的词汇表结构体来存储的

相关代码如下，在gpl.y中：

if(letter())

{

while(letter() || digit())

{

concat();

getch();

}

retract();

num=keyword();

if(num!=0)

{

printf("(%d, -) //KEYWORD\n", num);

return num; // return keyword

}

else

{

lexeme=malloc(strlen(token+1));

strcpy(lexeme,token);

yylval.string=lexeme;

wl[lenwl].p = (int\*)lexeme;

strcpy(wl[lenwl].m, token);

lenwl++;

printf("(%d, %p) //IDENTIFIER\n", IDENTIFIER, lexeme);

return IDENTIFIER;

}

}

if(digit())

{

while(digit())

{

concat();

getch();

}

retract();

lexeme=malloc(strlen(token+1));

strcpy(lexeme,token);

yylval.string=lexeme;

int \*p = malloc(4);

\*p = atoi(lexeme); // \*p is an int

wl[lenwl].p = (int\*)p;

strcpy(wl[lenwl].m, token);

lenwl++;

printf("(%d, %p) //INTEGER\n", INTEGER, p);

return INTEGER;

}

if(character=='"')

{

concat();

getch();

while(character!='"' && character!=EOF)

{

concat();

getch();

}

if(character==EOF)

{

printf("lex error: \" expected\n");

exit(1);

}

concat();

lexeme=malloc(strlen(token+1));

strcpy(lexeme,token);

yylval.string=lexeme;

wl[lenwl].p = (int\*) lexeme;

strcat(wl[lenwl].m, token);

lenwl++;

printf("(%d, %p) //TEXT\n", TEXT, lexeme);

return TEXT;

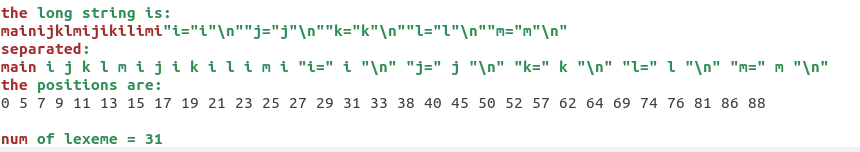
}

（5）实现一个存放字面量的字符串

已经实现

主要是用一个全局变量ls实现的

打印效果如图



（6）实现了哈希表

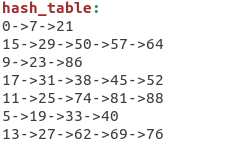
主要就是使用数据结构里面哈希表相关的指示

定义哈希表的大小为7

hash函数为：data % 7

这样就可以尽可能多的看到哈希表的效果

打印效果如图：



实验一所有功能都已实现